VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Kompresní algoritmy pro sady velmi podobných obrázků

# Compression Algorithms Suitable for Sets of Similar Images

2014 Marek Fajstl

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Zadání bakalářské práce

Student: **Marek Fajstl**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Kompresní algoritmy pro sady velmi podobných obrázků
Compression Algorithms Suitable for Sets of Similar Images

Zásady pro vypracování:

Cílem je najít vhodný algoritmus pro kompresi velkého počtu velmi podobných obrázků, lišících se jen v detailech. První částí práce bude prozkoumat již existující algoritmy a porovnat je jak z pohledu efektivity, tak kvality a rychlosti komprese. Druhou částí bude implementace vlastní varianty kompresního algoritmu, který bude co nejvhodnější pro dodanou sadu testovacích obrázků.

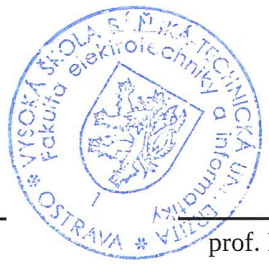
Seznam doporučené odborné literatury:

[1] SAYOOD, Khalid. Introduction to data compression. 3rd ed. San Francisco: Morgan Kaufmann Publishers, 2006, xxii, 680 s. ISBN 01-262-0862-X.
[2] PENNEBAKER, William B a Joan L MITCHELL. JPEG still image data compression standard. New York: Van Nostrand Reinhold, 1992, xviii, 638 p. ISBN 04-420-1272-1.
[3] OZER, Jan Lee. Video compression for flash, apple devices and html5. 1st ed. Galax, VA: Doceo Pub., 2011, p. cm. ISBN 978-097-6259-503.
[4] RICHARDSON, Iain E. The H.264 advanced video compression standards. 2nd ed. Chichester: John Wiley, 2003, xxx, 316 s. ISBN 978-0-470-51692-8.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Michal Holiš**

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2014




doc. Dr. Ing. Eduard Sojka
*vedoucí katedry*

prof. RNDr. Václav Snášel, CSc.
*děkan fakulty*

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2014

Marek Fajfr

.....

Rád bych poděkoval panu Ing. Michalu Holišovi za jeho cenné rady při psaní této bakalářské práce.

## Abstrakt

Tato bakalářská práce se zabývá kompresí dat a to zejména se zaměřením na komprimaci velmi podobných obrázků. Na začátku práce je uveden popis několika kódovacích technik, které jsou hojně využívané v různých kompresních metodách. V další části jsou již uvedeny komprese obrazu a to především komprese videa, která je přímo spjata s kompresí podobných obrázků. Popsané formáty videa jsou testovány na dodané sadě testovacích obrázků a porovnány jak z pohledu efektivity, tak kvality a rychlosti komprese. Poslední část je zaměřena na popis vlastní varianty kompresního algoritmu a celkové zhodnocení práce.

**Klíčová slova:** kompresní algoritmy, formáty videa, komprese podobných obrázků, JPEG, MPEG

## Abstract

This bachelor thesis deals with data compression with special focus on compression of very similar images. At the beginning of the thesis there are described several coding techniques, which are widely used in various compression methods. The next part describes image compression, especially video compression, which is directly related to compression of similar images. Described video formats are tested on a supplied set of images and then compared. The comparison focuses on performance, quality and speed of compression. The last part deals with a description of my own variants of compression algorithm and the overall evaluation of the work.

**Keywords:** compression algorithms, video formats, compression of similar images, JPEG, MPEG

## Seznam použitých zkratek a symbolů

| | | |
|---|---|---|
| RLE | – | Run-Length Encoding |
| LZW | – | Lempel-Ziv-Welch |
| LZ77 | – | Lempel-Ziv 77 |
| JPEG | – | Joint Photographic Experts Group |
| DCT | – | Discrete Cosine Transform |
| MPEG | – | Motion Pictures Experts Group |
| GOP | – | Group Of Pictures |
| WMV | – | Windows Media Video |
| SMPTE | – | The Society of Motion Pictures and Television Engineers |
| WHT | – | Walsh-Hadamard Transform |
| DWT | – | Discrete Wavelet Transform |

# Obsah

**1 Úvod** **4**

**2 Metody kódování** **5**
2.1 Run-Length Encoding (RLE) . . . . . 5
2.2 Huffmanovo kódování . . . . . 5
2.3 Shannon-Fanovo kódování . . . . . 6
2.4 Aritmetické kódování . . . . . 6
2.5 Lempel-Ziv-Welch (LZW) . . . . . 7

**3 Ztrátová komprese** **9**
3.1 JPEG . . . . . 9
3.2 MPEG . . . . . 10
3.3 Theora . . . . . 13
3.4 VP8 . . . . . 14
3.5 Windows Media Video . . . . . 14
3.6 Dirac . . . . . 14

**4 Bezeztrátová komprese** **15**
4.1 HuffYUV . . . . . 15
4.2 Lagarith . . . . . 15
4.3 FFV1 . . . . . 15

**5 Testování** **16**
5.1 Testování na základě stejné velikosti . . . . . 17
5.2 Testování na základě stejné kvality . . . . . 21

**6 Popis vlastní varianty kompresního algoritmu** **23**
6.1 Kodér . . . . . 24
6.2 Dekodér . . . . . 26
6.3 Dosažené výsledky . . . . . 27

**7 Závěr** **28**

**8 Reference** **29**

**Přílohy** **30**

**A Obsah CD** **31**

# Seznam obrázků


1 Příklad Huffmanova stromu . . . . . 6
2 Komprese JPEG . . . . . 10
3 Makroblok v systému 4:2:0 . . . . . 11
4 GOP . . . . . 11
5 Porovnání MPEG-2 s h.264 . . . . . 13
6 Původní obrázek z testovací sady . . . . . 16
7 Výřez původního obrázku z testovací sady . . . . . 17
8 MPEG-4 part 10 / H.264 main profile . . . . . 18
9 VP8 . . . . . 18
10 Theora . . . . . 18
11 Dirac . . . . . 19
12 MPEG-2 . . . . . 19
13 WMV2 . . . . . 19
14 Encoder . . . . . 23
15 Příklad rozdělení bloku 32 x 32 pixelů . . . . . 25
16 Komprese s tolerancí rovné sedmi . . . . . 27

# Seznam tabulek

1 Huffmanovo kódování . . . . . 6
2 Testování na základě stejné velikosti . . . . . 17
3 Testování na základě stejné kvality . . . . . 21
4 Kódování jednotlivých bloků . . . . . 24
5 Kódování barevné hodnoty obrázků . . . . . 26

# 1 Úvod

Pro uchování velkého množství dat je dobré použít nějakou kompresní metodu. Komprese je matematická metoda sloužící ke zmenšení velikosti souborů při zachování informace. Komprese mohou být různé, především musíme vzít v úvahu, která data se budou komprimovat. U ztrátové komprese obrazu, zvuku nebo videa se většinou vychází z nedokonalosti lidského vnímaní. Naopak u bezeztrátové komprese nesmí dojít k žádné ztrátě a komprimovaná data musí být zpětně rekonstruovatelná do přesné podoby dat vstupních. Tento druh komprese je vhodný například pro běžný text, kde nepřipadá v úvahu žádná ztráta na kvalitě.

Tato práce se zabývá kompresí obrázků, zachycených monitorovacím systémem Surface Scan System, který slouží k bezkontaktnímu monitorování, umožňujícího vyhodnocení vad na pásu plechu válcovaného za studena. Tento systém vyprodukuje velké množství podobných obrázků, které je třeba zkomprimovat a následně archivovat. Testovací sada obrázků, která je použita v této práci, je výstupem tohoto systému.

V první části této práce je popsáno několik kompresních metod, a to ztrátových i bezeztrátových. Větší důraz je ovšem kladen na kompresi videa, která je přímo spojena s komprimací podobných obrázků.

V další části práce jsou tyto kompresní metody otestovány na zadané sadě testovacích obrázků, jak z pohledu kompresního poměru, tak rychlosti komprese.

Poslední část práce je věnována popisu vlastní implementace kompresního algoritmu. Algoritmus byl navrhnut přímo pro dodanou sadu testovacích obrázků a je zde uvedeno porovnání s ostatními testovanými kompresními metodami.

# 2 Metody kódování

## 2.1 Run-Length Encoding (RLE)

Výhodou tohoto algoritmu je fakt, že je praktikovatelný na jakýkoliv druh dat. Záleží samozřejmě na jejich vlastnostech, od kterých se následně odvíjí kompresní poměr.

Tento algoritmus spočívá v tom, že pokud máme sekvenci stejných znaků za sebou, můžeme je zapsat jako jejich počet krát příslušný symbol (pixel). Tento způsob ukládání dat je aplikovatelný jak na textová, tak i obrazová data.

**Příklad:**

Vstup: AAAABBBAAAAAC
Po průchodu algoritmem RLE: 4A3B5A1C

Z příkladu je patrné, že tento princip bude dobře použitelný pro data, která mají na vstupu dlouhou posloupnost stejných znaků. Běžný text ovšem takovou vlastnost nemá, a proto je zřejmé, že tento algoritmus je vhodný na kódování grafických dat [6].

## 2.2 Huffmanovo kódování

Huffmanovo kódování je kompresní metoda, která patří mezi nejznámější ze skupiny algoritmů, které pracují na principu různého počtu znaků v kódovacích datech. Princip tohoto algoritmu spočívá v tom, že nejčastěji vyskytující se hodnoty se ukládají nejmenším počtem bitů. Naopak hodnoty, které se nejméně vyskytují, jsou ukládány dlouhými bitovými řetězci. Musí však platit, že žádný kód netvoří začátek jiného kódu [1].

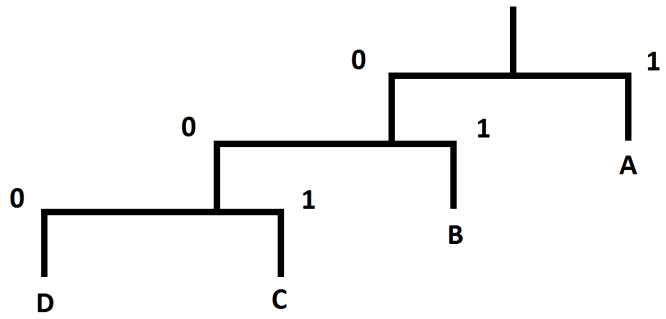
V následujícím příkladu pro vstupní symboly A, B, C, D, které jsou seřazeny podle počtu jejich výskytu, jsou uvedeny jejich počty a kódy, které je budou reprezentovat. Znaku s největším počtem výskytu "A"je přiřazen pouze jediný bit s hodnotou 1. V tomto případě tedy už žádný jiný kód znaku nemůže začínat jedničkou.

Dekomprese se provádí za pomocí dekódovacího stromu na obrázku č. 1. V tomto stromu platí, že každému symbolu původní zprávy odpovídá jeden list. Huffmanovo kódování není vhodné pro kompresi krátkých souborů, protože s každým komprimovaným souborem musíme přenášet i dekódovací strom o velikost 0,5 až 1 KB [1]. Huffmanovo kódování dosahuje optimálních výsledků za předpokladu, že jsou četnosti jednotlivých znaků mocninami čísla 2.

**Příklad:**

| Symbol | Počet | kód |
|---|---|---|
| A | 6 | 1 |
| B | 2 | 01 |
| C | 1 | 001 |
| D | 1 | 000 |

Tabulka 1: Huffmanovo kódování




Obrázek 1: Příklad Huffmanova stromu

## 2.3 Shannon-Fanovo kódování

Variantou Huffmanova kódování je kódování pomocí Shannon-Fanovy metody. Rozdíl těchto dvou kódovacích metod je v jiném způsobu vytváření značek kódu a v tom, že není třeba přenášet celou strukturu dekódovacího stromu, ale pouze informace o délkách jednotlivých značek [1].

## 2.4 Aritmetické kódování

Aritmetické kódování je také variantou Huffmanova kódování. Rozdíl těchto kompresí spočívá v tom, že každý vstupní symbol se kóduje na symbol o různém, ale ne celistvém počtu bitů. Je to umožněno tím, že celá vstupní zpráva je zakódována do jednoho racionálního čísla v intervalu od 0 do 1. Uvedu jednoduchý příklad: Pokud máme text se symboly A, B a C, použijeme pro jejich zakódování 4 bity. Pro A - 00, B - 01, C - 10. Jedna možnost a to 11 zůstane nevyužitá a právě toto kódování ji využije. Místo toho abychom použili dvojkovou soustavu, použijme soustavu trojkovou, tedy každý znak bude odpovídat jednomu číslu [1].

**Příklad:**
Vstup: AACBABB
Po průchodu aritmetickým kódováním získáme následující číslo v trojkové soustavě $0.0021011_3$.

Poté se toto číslo převede do počítačové dvojkové soustavy, a může být tak uloženo v paměti počítače. Toto kódování sice řeší problém předchozích dvou, ale na druhou stranu potřebuje větší výpočetní výkon a při větších souborech se musí dělit na jednotlivé části kvůli tomu, že desetinná místa čísla ohromně vzrostou už pro malé vstupní soubory.

## 2.5 Lempel-Ziv-Welch (LZW)

V roce 1977 Abraham Lempel vytvořil algoritmus, který byl rok poté vylepšen Jacobem Zivem [5]. Dnes se proto setkáváme s algoritmy pojmenovanými podle těchto dvou pánů LZ77 a LZ78. V názvu je zmíněno ještě jméno Terryho Welche, který přispěl v roce 1984 k dalšímu vylepšení předchozích dvou tím, že algoritmus upravil pro potřeby hardwarových řadičů. A tak vznikla jedna z nejrozšířenějších komprimačních metod, kterou používají různé aplikace pro komprimaci textu i obrazu. Běžný anglický text dokáže zkomprimovat asi na polovinu původní délky.

### 2.5.1 LZ77

LZ77 funguje tak, že pracuje se dvěma částmi textu zároveň. První je aktuální okno, které komprimuje, a druhé posuvné okno (sliding window) [6]. V posuvném okně se kompresní algoritmus snaží nalézt co nejdelší podřetězec odpovídající řetězci na vstupu. Pokud se to podaří, vytvoří se na něj v aktuálním okně odkaz. Tento odkaz musí obsahovat ukazatele na začátek podřetězce a délku podřetězce.

**Příklad:**
Vstup: Leze po železe
Po průchodu algoritmem LZ77: Leze po že[10,4]

*Pozn.*
Tento zápis je pouze schématický. Jedná se o to, aby dekodér z posledních deseti znaků vybral první čtyři.

Tento princip je velice podobný jako princip u hypertextových odkazů, které jsou běžně používány na webových stránkách. Jeden z hlavních problémů je velikost posuvného okna (sliding window), protože toto okno musí být dostatečně velké, aby se podařilo najít požadovaný podřetězec a přitom dostatečně malé, aby komprese netrvala příliš dlouho.

Většinou se tento problém řeší tak, že se nechá na uživateli volba podmínek komprese. Velikost okna se pohybuje řádově v kilobajtech.

### 2.5.2 LZ78

Zatímco metoda LZ77 používá posuvné okno(sliding window), metoda LZ78 si tvoří vlastní slovník, do kterého zapisuje již nalezená spojení [6]. Při čtení souboru tedy přečte řetězec a pokud je tento řetězec obsažen ve slovníku, nahradí ho příslušným odkazem, v opačném případě ho připíše do slovníku. Tato metoda je velice paměťově náročná, protože pro objemné vstupy může slovník zabrat celou volnou paměť. Tento problém lze řešit několika způsoby. Jedna z metod pro uvolnění paměti spočívá v tom, že jakmile dojde k nedostatku paměti, slovník se zmrazí, smaže a začne se vytvářet nový.

# 3 Ztrátová komprese

## 3.1 JPEG

Asi nejznámější formát obrázků, který využívá ztrátovou kompresi je formát JPEG (Joint Photographic Experts Group) , což je komise pro standardy pracující pod International Standard Organization (ISO) [1]. JPEG není přesně definovaným grafickým formátem, ale jedná se spíše o sadu kompresních metod, které mohou být přizpůsobeny konkrétním požadavkům uživatele.

JPEG komprimuje obrazy s 24 bitovou hloubkou(true color) [1]. U JPEG není použit klasický model barev RGB ale YCbCr, který obsahuje jasovou složku Y a dvě barevné složky Cb a Cr. Model barev YCbCr, který má jasovou složku oddělenou od barevných, je pro ztrátovou kompresi podstatně výhodnější. Lidské oko je podstatně méně citlivější na změny barev než na změny jasu. Proto lze u barevné složky obrazu připustit větší ztrátu informace, než u jasové složky. Tímto se dosáhne celkově vyššího kompresního poměru [7].

**Postup komprese je následující:**

1. Převedení obrázku do barevného modelu YCbCr.
2. Snížení přesnosti informací o barvě (podvzorkování).
3. Rozdělit obrázek na čtvercové bloky po 8 x 8 bodech, kde se pro každý blok provede diskrétní cosinová transformace (DCT). Cílem je oddělení informací s nízkými frekvencemi (pomalých změn jasu nebo barev) od vysokých frekvencí (náhlých změn jasu a barev).
4. Provede se kvantizace, kde dochází ke zmenšení hodnot a především malé hodnoty se vlivem kvantizace úplně zanedbají => ztrátová komprese.
5. Výsledná data v bloku se vyčtou metodou "cik - cak"a komprimují bezeztrátově pomocí Run-length encoding (RLE). Tato data se následně uloží na výstup Huffmanovým nebo aritmetickým kódováním.

JPEG je velice vhodný pro kompresi fotografií s velkou barevnou hloubkou. U takových grafických předloh dosahuje kompresní poměr 20:1 bez okem rozpoznatelné újmy na kvalitě obrázku. Naopak při práci s vektorovými obrázky a jednoduchou grafikou je vhodné použít jiný formát.

Výhodou je nastavitelný Q faktor udávající kvalitu obrázku po kompresi. Čím menší je hodnota Q faktoru, tím menší je velikost výsledného souboru a pochopitelné i horší kvalita obrázku.

Obrázek 2: Komprese JPEG

Na obrázku 2 vidíme, jak kompresní poměr vzrůstá zleva doprava. V pravé části obrázku jsou patrné již zmíněné čtverce o rozměrech 8 x 8 pixelu, což je způsobeno ztrátovostí dat u JPEG a nastavením příliš nízkého Q faktoru.

## 3.2 MPEG

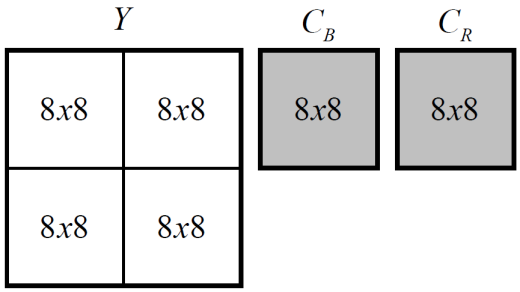
MPEG je zkratka z anglických slov(Motion Pictures Experts Group), což v češtině znamená "Skupina expertů pro pohyblivý obraz". Tato skupina se zabývala kompresí videa a snažila se jej standardizovat [8].

### 3.2.1 MPEG-1

Tento formát byl dokončen v roce 1991. Kodek pracuje s velikostí obrazu 352x288 pixelových bodů a rychlostí 25 snímků za sekundu při datovém toku do 1,5 Mbit/s [8]. MPEG-1 byl primárně určený pro multimediální aplikace, které vyžadují náhodný přístup k videu tzn. rychlý posun vpřed, vzad, hledání času, nebo pozdější manipulace s videem. Ovšem hlavním požadavkem bylo zkomprimovat velikost datového toku.

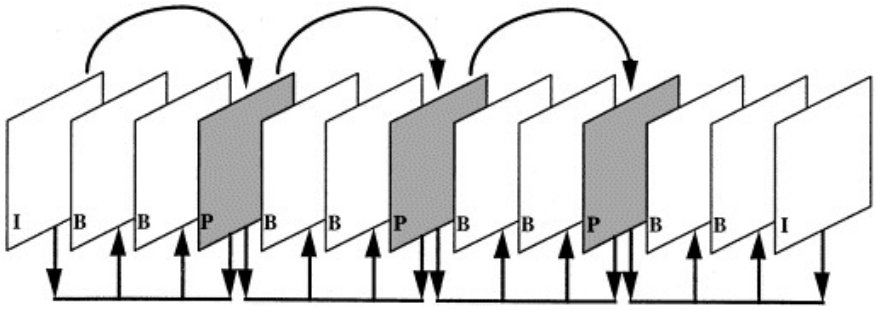
**Popis algoritmu:**
MPEG ukládá 3 druhy snímků, které jsou organizovány ve skupinách GOP (Group Of Pictures) obr. č. 4. Základní jednotkou kódování uvnitř obrázku je makroblok. Makroblok je tvořen čtyřmi jasovými bloky a dvěma chrominančními bloky, přičemž každý blok je reprezentován 8 x 8 bitů. Počet bloků v makrobloku záleží na způsobu vzorkování.

Nejčastější způsob je označován 4:2:0 (čtyři jasové bloky, dva chrominanční bloky $C_B$ a $C_R$) obr. č. 3




Obrázek 3: Makroblok v systému 4:2:0

**I(Intraframes - klíčové)** - snímky, které jsou uloženy kompletní. Pouze jsou kódovány pomocí DCT, bez použití informací z ostatních obrázků - jako u standardu JPEG. Od těchto obrázků se odvíjí následující dva.
**P(Predicted - předpokládané)** - snímky kódované pouze jednosměrně tzn. ,že aktuální snímek P je zakódovaný pomocí předcházejícího snímku typu I nebo B. Tyto P snímky se ukládají pouze jako rozdíl od předchozího snímku => vyšší kompresní poměr
**B(Bidirectional - obousměrné)** - snímky kódované obousměrně tzn. ,že aktuální snímek je kódován pomocí předcházejícího snímku a zároveň snímku následujícího. Pro toto obousměrné kódování se používají jak snímky typu I, tak snímky typu P. Tento druh snímků má nejvyšší kompresní poměr.




Obrázek 4: GOP

### 3.2.2 MPEG-2

Skupina MPEG pokračovala ve své standardizaci a v roce 1994 přišla s novým standardem MPEG-2, který přímo navazuje na předchozí práci. V principu se kódování nezměnilo, pouze byla přidána určitá rozšíření. MPEG-2 obsahuje různé profily a úrovně, kde kombinací jednotlivých parametrů, jako například rozlišení, vzorkovací frekvence,volitelná

přesnost DCT a počet snímků za vteřinu. Také na rozdíl od MPEG-1 podporuje 5.1 kanálový zvuk. MPEG-2 se používá pro kódování videa na DVD a pro televizní vysílání [10].

Každý dekodér musí zvládnout dekódování všech datových toků až do maximálního toku pro danou úroveň a daný profil. Vlastnosti kodéru nejsou přesně definovány pro možnost jejich dalšího vývoje. Na druhou stranu je velký důraz kladen na to, aby postupné zdokonalování kodérů se obešlo bez změny dekodérů, které by měli být zejména při větších výrobních sériích dostupné. Další velkou výhodou je fakt, že jakékoliv video zakódované v MPEG-1 je možno dekódovat v jakémkoli dekodéru MPEG-2 [10].

### 3.2.3 MPEG-4

MPEG-4 je standart pro kompresi audiovizuálního signálu, který byl představen v roce 1998 a je nástupcem MPEG-2. MPEG-4 obsahuje mnoho částí, z nichž některé se ještě stále vyvíjí a tak uvedu pouze nejpoužívanější z nich.

**MPEG-4 part 2**
V roce 1999 byl standardizován MPEG-4 part 2, který má 21 profilů. Nejpoužívanější profily jsou SP (Simple Profile) a ASP (Advanced Simple Profile). SP je určený pro nízké datové toky nebo nízké rozlišení na mobilních telefonech nebo průmyslových kamerách. Profil ASP oproti předchozímu podporuje prokládané video, obousměrně kódované snímky (B-frame), GMC a QPel.

**GMC** (Global Motion Compensation) - makrobloky objektu videa mohou mít stejný pohyb, a tak je možno jej nahradit jedním společným vektorem. Například, když kamera "zoomuje"na scénu, nebo se provádí rotace celé scény.

**PeL** (Quarter-pixel motion) - je metoda detekce pohybu mezi dvěma rámci, která přináší ostřejší obraz. QPel je nástupcem HalfPel, která nepodává tak ostrý obraz, za to je méně náročná na výpočet.

*Pozn.* Více o těchto algoritmech se dozvíte v knihách [12] a [13].

**MPEG-4 part 10 / H.264**

MPEG-4 part 10 nebo také H.264 je kompresní formát, který je momentálně nejhojněji používaným formátem pro distribuci videa. Byl standardizován v roce 2003 skupinou VCEG (Video Coding Experts Group) za spolupráce MPEG (Moving Pictures Experts Group). Cílem tohoto projektu bylo dosažení vyšší kvalitě videa při nižší přenosové rychlosti. K tomu využívá řadu vylepšení oproti předchozím a to např. CAVLC (Context-

Adaptive Variable-Length Coding) a CABAC (Context-Adaptive Binary Arithmetic Coding), což jsou bezeztrátové algoritmy entropického kódování [14], [15].

Na obrázku č. 5 je vlevo vidět komprese pomocí MPEG-2 s datovým tokem 10 Mb/s, kdežto vpravo byl použit kompresní formát H.264 (Main profile) s datovým tokem pouze 2 Mb/s. Tohle je ovšem vysoce pohyblivá scéna, u statických scén není rozdíl až tak markantní. Obecně se udává padesáti procentní úspora, ale samozřejmě na úkor vyššího výpočetního výkonu, který je třeba jak pro kódování tak dekódování videa [12].


Obrázek 5: Porovnání MPEG-2 s h.264

## 3.3 Theora

Theora je svobodný formát videa, který byl vyvinut nadací Xiph.Org Foundation na základě formátu VP3 firmy On2 Technologies. Ačkoli tato firma stále vlastní některé patenty na tento formát, dala svolení k jejich volnému užívání. Theora je hodně podporovaná společnostmi jako je Google a Mozilla a to hlavně pro její volnou šiřitelnost. Nevýhodou ovšem i nadále zůstává nižší kvalita komprimovaného videa, než u hojně používaného MPEGu-4.

Theora využívá mnoho stejných nebo podobných kompresních algoritmů jako MPEG. V současnosti podporuje základní chrominační vzorkovací formáty Y'CbCr (4:2:0, 4:2:2 a 4:4:4). Jsou zde pouze dva druhy snímku a to I-snímek a P-snímek. Chybí zde B-snímky jako je tomu u např. MPEGu-4 AVC [17].

Theora také rozděluje obraz na bloky o velikosti 8x8 obrazového bodu, na super bloky a podobně. Stejně jako většina ztrátových kompresních metod provádí také diskrétní kosinovou transformaci [17].

## 3.4 VP8

VP8 je další video formát z produkce On2 Technologies, který byl vyvinut v roce 2008. On2 Technologies byla v roce 2010 odkoupena společností Google, který představil projekt WebM. Cílem tohoto projektu bylo stát se konkurentem formátu H.264 a nahradit ho především na webu. Od spuštění projektu se počet podporovatelů VP8 zvětšuje, jako příklad můžeme uvést YouTube, Skype a také většina internetových prohlížečů [18].

Komprese VP8 je v podstatě založena na principech MPEG-2 a H.264. Využívá se zde kromě diskrétní cosinové transformace (DCT) také Walsh-Hadamardovu tranformace (WHT), která využívá korelaci mezi DC koeficienty a snižuje tak redundanci [23].

## 3.5 Windows Media Video

Windows Media Video (WMV) je video formát vyvíjen společností Microsoft, který funguje podobně jako MPEG-4, ovšem nedosahuje takových výsledků jako zmíněný MPEG-4. Tento video formát se postupně objevoval ve verzích WMV1, WMV2, a WMV3, někdy taky označovaném číslicemi 7,8,9 podle verzí windows media playeru [19].

V roce 2006 byl standardizován skupinou SMPTE (The Society of Motion Pictures and Television Engineers) jako otevřený standart VC-1, který také tvoří jeden z formátů pro Blu-ray Disc, HD DVD a taktéž se s ním můžeme setkat u herní konzole Xbox 360 [2].

## 3.6 Dirac

Dirac je video formát vyvíjený společností BBC pojmenovaný podle britského herce Paula Diraca. Na rozdíl od většiny kompresních formátů, které jsou založeny na dikrétní kosinové transformaci (DCT), dirac využívá vlnkovou transformaci (DWT) [20]. Tato transformace například umožňuje přesnější určení bitového toku kódovaného videa.

Dirac Pro který je zástupcem z této rodiny byl standardizován skupinou SMPTE jako otevřený standard VC-2 a byl nasazen společností BBC při vysílání letních olympijských her roku 2008 v Pekingu.

# 4 Bezeztrátová komprese

## 4.1 HuffYUV

HuffYUV je jeden z nejpoužívanějších bezeztrátových formátů videa, jehož autorem je Ben Rudiak-Gould. Velkou výhodou tohoto algoritmu je vysoká rychlost komprese a volná šiřitelnost. Původní velikost souboru dokáže zkomprimovat na hodnoty kolem 40 %. Funkce formátu HuffYUV spočívá v predikování obrazových snímků a rozdílovou chybu ukládá Huffmanovým kódováním [21].

## 4.2 Lagarith

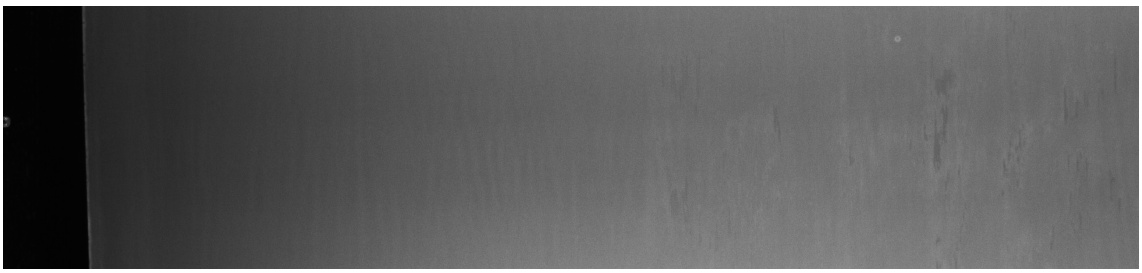
Lagarith je také volně šiřitelný formát, který je nástupcem výše zmiňovaného formátu HuffYUV. Tvůrcem tohoto způsobu kódování obrazu je Ben Greenwod. Ačkoli Lagarith nedosahuje takové rychlosti komprese jako HuffYUV má oproti svému předchůdci jinou výhodu. Bezeztrátové formáty jsou vhodné při manipulaci s videem (např. při střihu), protože u nich nedochází k degradaci kvality jako u ztrátových kompresních metod. Jelikož Lagarith kóduje každý obrázek zvlášť, je tato manipulace s videem snadnější (nemusejí se dopočítávat obrázky z klíčových snímků). Účinnost komprese typicky dosahuje o 30 % lepších výsledků než u jeho předchůdce HuffYUV [21].

## 4.3 FFV1

FFV1 je nejnovější ze zmíněných bezeztrátových formátů, který se pohybuje z hlediska účinnosti a rychlosti komprese někde mezi formáty HuffYUV a Lagarith. Jeho implementace je zahrnuta v projketu FFmpeg a je dostupný na více platformách než Lagarith, který je určen pouze pro Windows [21].

# 5 Testování

Pro testování všech algoritmů byla využita testovací sada obrázků, která byla zachycena systémem Surface Scan System. Tento bezkontaktní monitorovací systém umožňuje vyhodnocení vad na pásu plechu válcovaného za studena. Cílem systému je monitorovat kvalitu výroby v reálném čase, analyzovat a archivovat vyhodnocená data. Sufrace Scan System vyhodnocuje povrchové vady pásu a je dodáván včetně produktu Shape Scan System, který analyzuje rovinnost vyráběného pásu. Oba produkty jsou vyvinuty na principu snadné integrace do výrobního procesu [22].



Obrázek 6: Původní obrázek z testovací sady

Obrázky o rozlišení 2592 x 600 s osmibitovou barevnou hloubkou (pouze odstíny šedi), jsou již komprimovány standardem JPEG. Z tohoto důvodu nebyly při testování použity bezeztrátové algoritmy, protože žádný z bezeztrátových algoritmů nemůže konkurovat (z hlediska kompresního poměru) ztrátovým.

Jeden obrázek má v průměru velikost 127 000 bajtů. Tato sada obsahuje sto testovacích obrázků a její velikost je 12 718 795 bajtů. Z této hodnoty budu vycházet u stanoveni kompresního poměru, což je podíl velikosti nekomprimovaných (vstupních) dat ku velikosti komprimovaných (výstupních) dat.

$$\text{kompresní poměr} = \frac{\text{původní velikost}}{\text{komprimovaná velikost}}$$

**FFmpeg**

Pro všechny následující testy byla použita kolekce svobodného softwaru FFmpeg. Tento projekt založený Fabriciem Bellardem umožňuje nahrávání, konverzi a streamování zvukových a obrazových dat. Především díky knihovně libavcodec dokáže kódovat a dekódovat široké spektrum audio-video formátů. Tento program je vhodný i na kompresi výše zmíněných obrázků do určitého formátu videa.

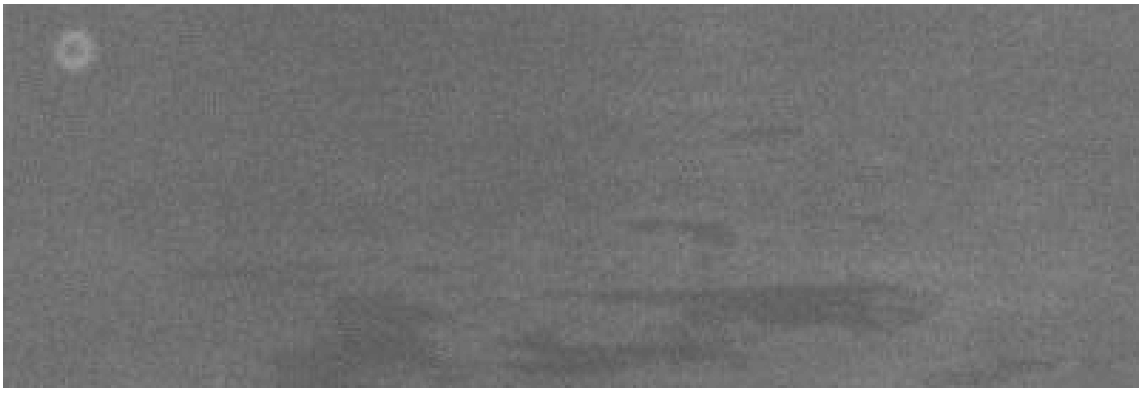
## 5.1 Testování na základě stejné velikosti

Pro názornou ukázku kvality obrazu jsem zvolil výřez jednoho z testovaných obrázků (obrázek č. 7), který je specifický svou stopou. V následujících obrázcích je vidět stejný výřez komprimovaný několika ztrátovými kompresními metodami. Z důvodu objektivity jsem se snažil o co možná nejmenší velikostní rozdíl mezi jednotlivými metodami, které jsou uvedeny v tabulce č. 2.

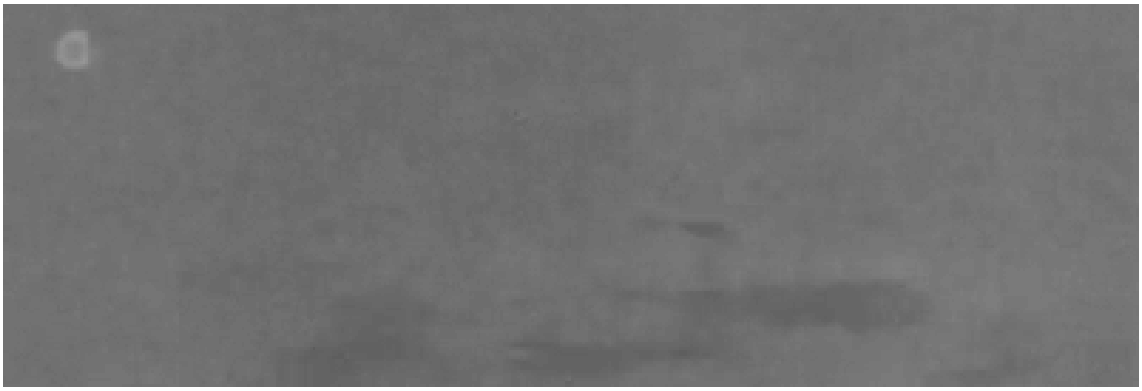
Jelikož jsou testovací obrázky hodně tmavé a při tisku by nebyly dobře rozpoznatelné rozdíly jednotlivých kompresních metod, přidal jsem všem zde uvedeným obrázkům o sto procent větší kontrast a o sedmdesát procent větší jas. Neupravené obrázky spolu s ostatními kompresní metodami, které zde nejsou zařazeny najdete na přiloženém CD i s celou původní sadou testovacích obrázků.

| **Formát** | **Komprimovaná velikost [B]** | **Kompresní poměr** | **Čas [s] komprese** |
|---|---|---|---|
| Dirac | 791 152 | 16,08 | 35,1 |
| H.264 high profile | 891 135 | 14,27 | 12 |
| H.264 main profile | 885 969 | 14,36 | 12 |
| MPEG-2 | 765 034 | 16,63 | 3 |
| MPEG-4 ASP | 832 751 | 15,27 | 3 |
| Theora | 796 047 | 15,98 | 16,8 |
| VP8 | 850 370 | 14,96 | 29,8 |
| WMV1 | 860 023 | 14,79 | 3,5 |
| WMV2 | 810 150 | 15,7 | 3,2 |

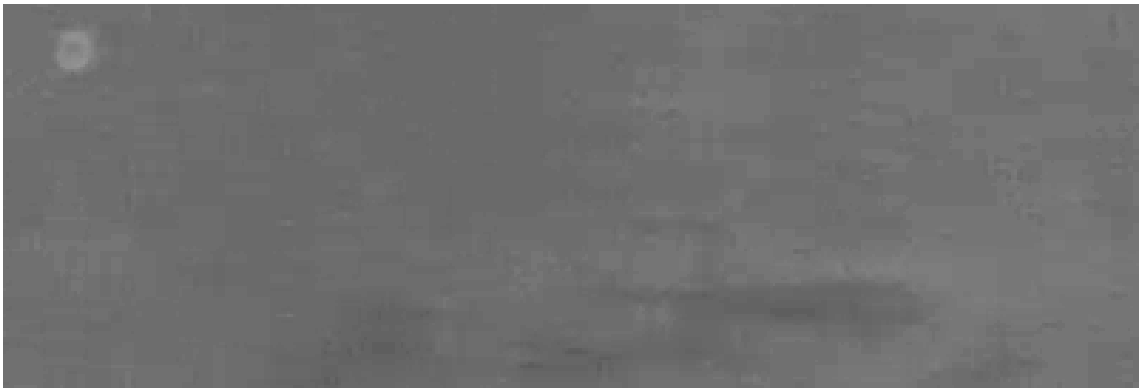
Tabulka 2: Testování na základě stejné velikosti



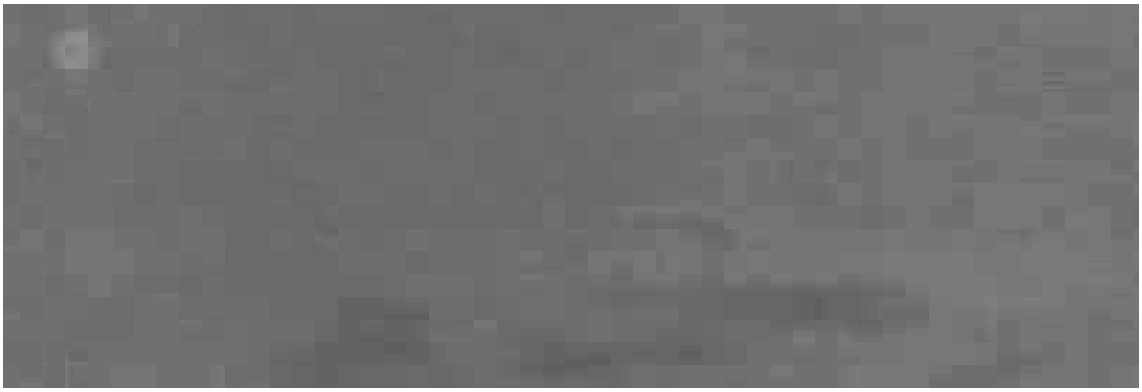
Obrázek 7: Výřez původního obrázku z testovací sady



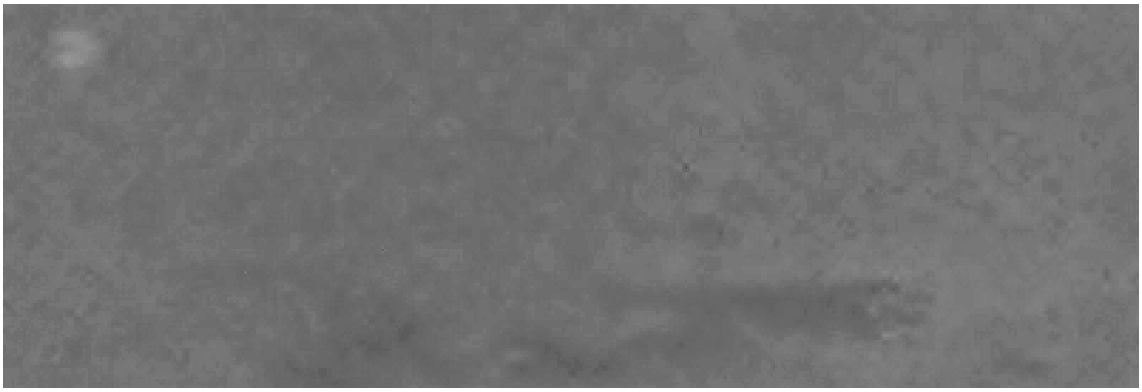
Obrázek 8: MPEG-4 part 10 / H.264 main profile



Obrázek 9: VP8



Obrázek 10: Theora



Obrázek 11: Dirac

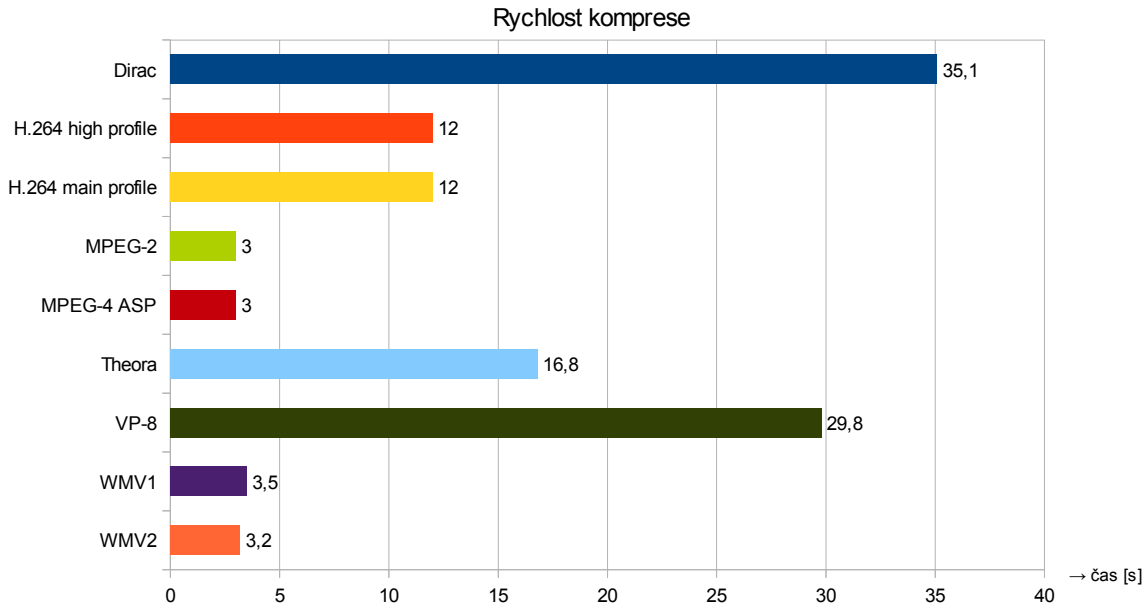
Obrázek 12: MPEG-2

Obrázek 13: WMV2

V tomto testu nejvíce vynikal formát H.264, který svými vyhlazenými obrázky zachovával i velké množství detailů. Při porovnání dvou profilů tohoto formátu se o trochu lepší jevil main profile, protože u high profilu byla patrná větší míra vyhlazování, a tudíž i nějaká ztráta informace. Podobných výsledků dosáhl i formát VP8, který až na pár míst kde jsou rozpoznatelné čtverce, vypadá velice dobře.

U Theory jsou prakticky po celém obrázku vidět jednotlivé čtverečky, ovšem stále je rozpoznatelná původní specifická stopa, a proto je tento obrázek stále použitelný. Zajímavých výsledků dosáhl Dirac, kde oproti ostatním metodám nedochází při degradaci kvality k nárůstu počtu viditelných čtverců, ale k určitému zrnění obrazu. V porovnání kvality je Dirac srovnatelný s Theorou.

Ostatní testované formáty MPEG-2, MPEG-4 AVC, WMV1 a WMV2, skončili v tomto testu na posledním místě. U obrázků komprimovanými těmito metodami je jen těžko patrná původní stopa a při kompresním poměru rovné plus mínus patnácti, jsou nevhodné pro testování takovýchto obrázků.

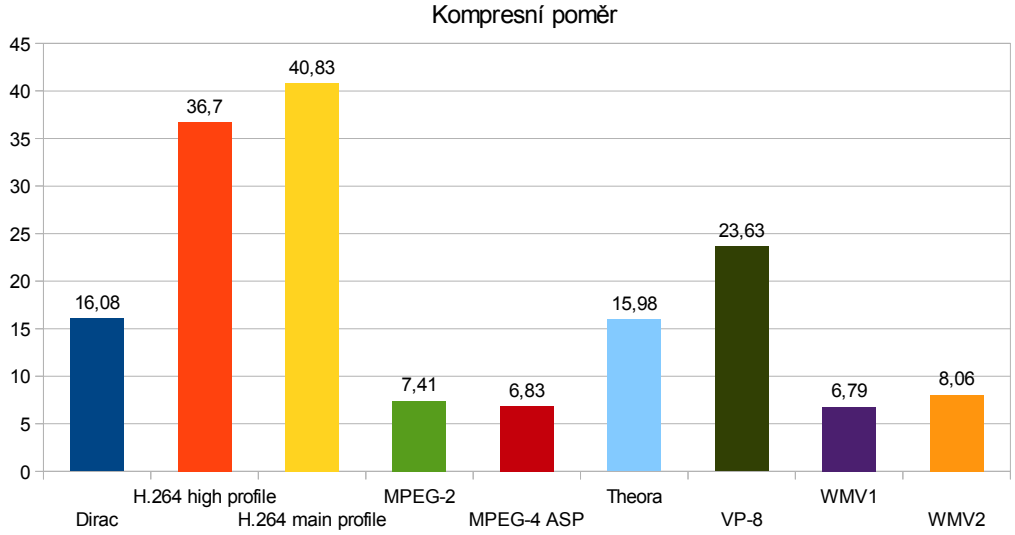


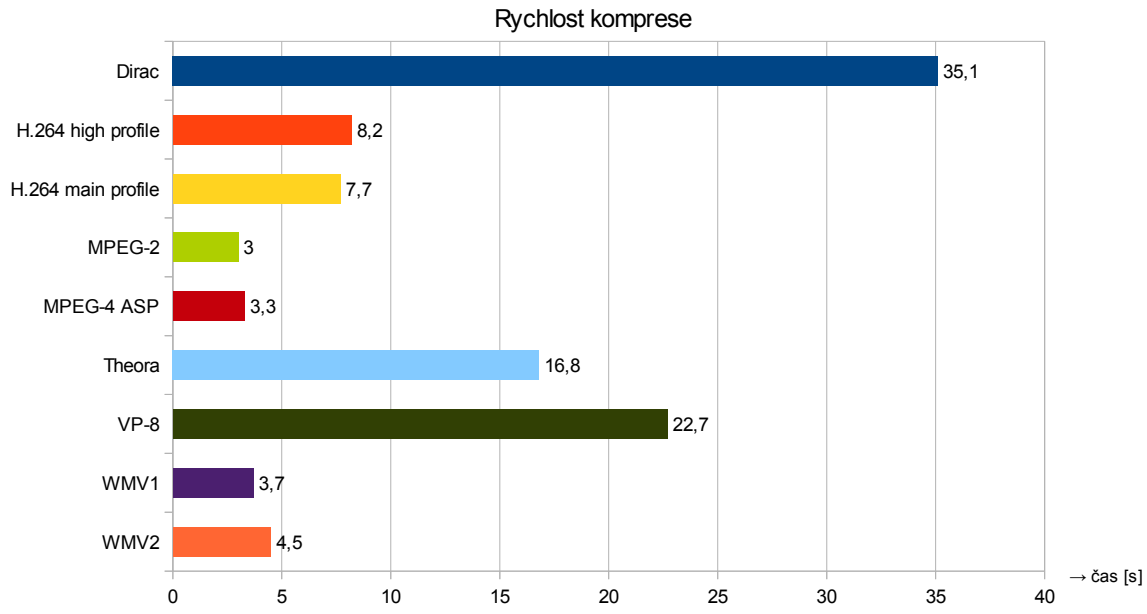
## 5.2 Testování na základě stejné kvality

V této části jsou uvedeny kompresní metody videa, které jsou porovnány na základě stejné kvality jednotlivých obrázků. V následujícím grafu a tabulce č. 3 jsou vidět komprimované velikosti testovací sady obrázků pro dostačující kvalitu k archivaci.

| Formát | Komprimovaná velikost [B] | Kompresní poměr | Čas [s] komprese |
|---|---|---|---|
| Dirac | 791 152 | 16,08 | 35,1 |
| H.264 high profile | 346 552 | 36,7 | 8,2 |
| H.264 main profile | 311 478 | 40,83 | 7,7 |
| MPEG-2 | 1 716 996 | 7,41 | 7,41 |
| MPEG-4 ASP | 1 861 728 | 15,27 | 6,83 |
| Theora | 796 047 | 15,98 | 16,8 |
| VP8 | 538 259 | 23,63 | 22,7 |
| WMV1 | 1 873 905 | 6,79 | 3,7 |
| WMV2 | 1 578 818 | 8,06 | 4,5 |

Tabulka 3: Testování na základě stejné kvality






*Pozn.* Všechna uvedená měření byla prováděna na stroji Intel (R) Core (TM) Duo CPU P8400 (2.26 GHz, 3MB cache, 1066MHz FSB), RAM 4,00 GB DDR II, pevný disk 500 GB 7200 ot./min a grafickou kartou ATI Mobility Radeon HD4670 (512MB).

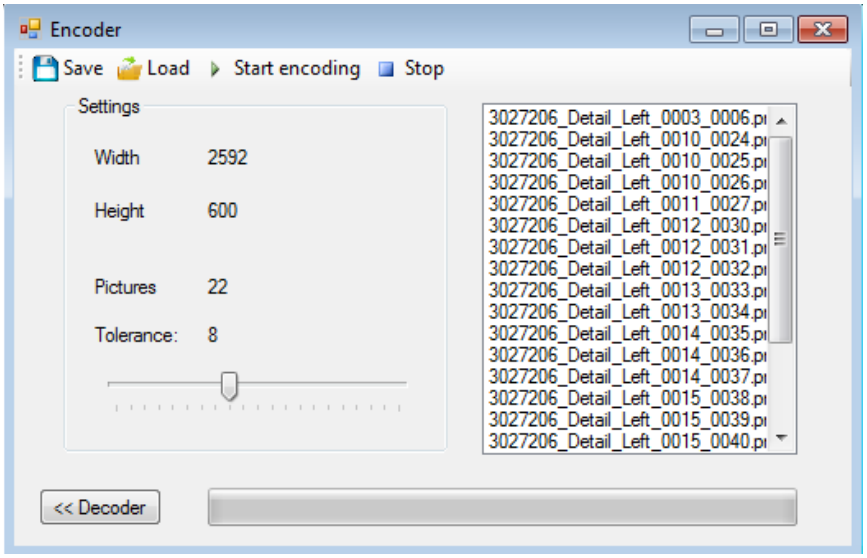
# 6 Popis vlastní varianty kompresního algoritmu

Pro naprogramování tohoto kompresního algoritmu jsem použil programovací jazyk C# a jako vývojové prostředí Microsoft Visual studio 2010. Můj kompresní algoritmus je ztrátový tak, že sjednocuje vetší plochy o malé barevné hloubce a ukládá je do bloků o různé velikosti.

Program je naprogramován pouze pro kompresi černobílých osmibitových obrázku a je prioritně určen pro dodanou sadu testovacích obrázků, které jsou popsané v předchozí kapitole.

Osmibitový černobílý obrázek je složen z pixelů, které mohou nabývat hodnot 0-255. Nula představuje sytě černou barvu a naopak nejvyšší možná hodnota 255 bílou barvu. Zbylé hodnoty jsou přirozeně odstíny šedi. Čím vyšší hodnota, tím světlejší odstín.

Před samotným spuštěním komprese si uživatel může zvolit míru komprese pomocí jezdce, který je umístěn v hlavním okně při spuštění programu (obrázek č. 14). Tato volba udává míru tolerance, která určuje, jestli se blok rozdělí na menší bloky nebo se zakóduje jako jeden velký blok. Například pokud je tolerance nastavena na hodnotu 8 a jednotlivé pixely v bloku nabývají hodnot od 42 do 51, rozdíl těchto mezních hodnot je větší než tolerance, tudíž se blok rozdělí.




Obrázek 14: Encoder

## 6.1 Kodér

1. V první řadě se provede rozdělení obrázku na bloky o velikosti 32 x 32 pixelů, kde se kontroluje každý pixel zvlášť a při větší barevné odchylce než je dovoleno v toleranci, se tento blok rozdělí na 4 bloky o velikosti 16 x 16 pixelů. Pokud je barevná hloubka bloku v toleranci, uloží se tento blok s průměrnou barvou všech kontrolovaných pixelů.

2. První krok se opakuje i pro rozdělené menší bloky (vždy se rozdělí na 4 stejně velké čtvrtiny bloku) až do velikosti 2 x 2 pixelů. Takto velký blok o čtyřech pixelech je nejmenší možný a z tohoto důvodu se bez ohledu na barevnou hloubku jednoduše spočítá průměr hodnot, který pak reprezentuje výslednou barvu tohoto bloku. Příklad rozdělení bloku je vidět na obrázku č. 15.

3. Z výše uvedeného rozdělení do bloků se musí přirozeně zapisovat informace o jaký blok se vlastně jedná. Pro kódování velikosti bloků jsem použil jednoduché huffmanovo kódování (tabulka č. 4).

| **Velikost bloku** | **binární reprezentace** |
|---|---|
| 32 x 32 | 110 |
| 16 x 16 | 10 |
| 8 x 8 | 01 |
| 4 x 4 | 00 |
| 2 x 2 | 111 |

Tabulka 4: Kódování jednotlivých bloků

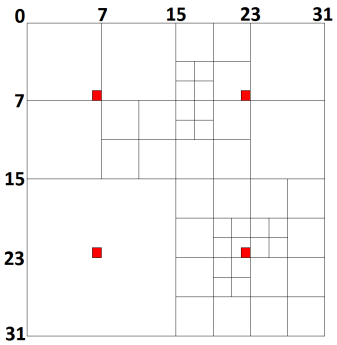
Z tabulky č. 4 je patrné, že největší a nejmenší blok je kódován nejdelším řetězcem bitů. Pro blok 32 x 32 pixelů je to vcelku pochopitelné, protože takto velkých bloků bude i při vyšší toleranci nejméně. Naopak u nejmenšího možného bloku o rozměrech 2 x 2 pixelů, se především u nižší tolerance dá předpokládat poměrně silné zastoupení. Ovšem jelikož je to nejmenší možný blok, můžeme se stoprocentní jistotou tvrdit, že za prvním blokem o zmíněné velikosti budou následovat další tři stejné bloky. Tímto lze vypustit informaci o velikosti těchto třech následujících bloků a ušetřit tak další cenné místo na disku.

4. Kódování se provádí napříč všemi obrázky následovně:

   Začíná se v levém horním rohu, kde se vytvoří blok 32 x 32 pixelů, rozdělí se podle bodu 1, 2 a zakóduje do výsledného binárního souboru. Následně se pokračuje, nikoli dalším blokem téhož obrázku, ale dalším obrázkem o stejných souřadnicích a velikosti. Této vlastnosti se využívá při predikci obrázků předchozího na následující, která je zmíněna níže.

Po zakódování prvního bloku u všech obrázků se přejde na další blok, umístěný pod předchozím blokem (posunutý po ypsilonové souřadnici o 32 pixelů). V momentě, kdy kompresní algoritmus narazí na dolní mez ypsilonové souřadnice, se za ideálních podmínek posune po x-ové souřadnici a pokračuje se dál. Ovšem pokud výška obrázku není dělitelná číslem 32, jako je tomu u dodané sadě testovacích obrázků (2592 x 600), nabízejí se dvě metody řešení. První spočívá v doplnění zbývajících 24 pixelů (18 bloků 32 x 32 dává dohromady 576) menšími bloky. Já jsem ovšem zvolil druhou metodu, kde se jednoduše posledních 32 x 8 pixelů ignoruje a jsou zakódovány znovu v příslušných blocích jako je tomu u ostatních částech obrázků (tato metoda je efektivnější pro zadanou sadu testovacích obrázků).

5. Pro větší kompresní poměr je samozřejmě využita podobnost mezi jednotlivými obrázky. Jak už bylo zmíněno v předchozím textu jednotlivé bloky jsou kódovány za sebou tak jak jdou po sobě jednotlivé obrázky a tato vlastnost přímo nahrává využití předchozího obrázku pro ten následující. Tohoto je docíleno čtyřmi klíčovými body v bloku se souřadnicemi: [7,7], [7, 23], [23, 7], [23, 23]. Na obrázků č. 15 jsou červeně zbarvené tyto čtyři klíčové body, z kterých se vychází při kódování následujícího obrázku. Každý z těchto bodů se využívá pro čtvrtinu celého bloku, tedy pro 16 x 16 pixelů.




Obrázek 15: Příklad rozdělení bloku 32 x 32 pixelů

Z těchto klíčových bodů předchozího obrázku jsou poté odvozeny následující obrázky a to tak, že se vždy odečte hodnota barvy předchozího příslušného klíčového bodu od kódované průměrné hodnoty barvy bloku.

$$\text{kódovaná hodnota} = \text{barva kódovaného bloku} - \text{barva klíčového bodu}$$

Jelikož předpokládáme, že tyto hodnoty budou plus mínus stejné (z důvodu podobnosti obrázků), bude se výsledná kódovaná hodnota pohybovat kolem nuly.

Pro tento jev jsem využil huffmanovo kódování, kde se nejčastěji využívaná hodnota kóduje nejkratším řetězcem bitů. Výsledkem tedy bude místo osmibitové hodnoty, která reprezentuje jednu barvu ze šedé škály, hodnota z tabulky č. 5.

Samozřejmě ne vždy budou ideální podmínky a z tohoto důvodu musí být uveden i příslušný znak pro zakódování barevné informace klasickým způsobem. Tento znak má binární reprezentaci 111 a za ním pak následuje 8 bitů, které jsou vyhrazeny pro nekomprimovanou barevnou hodnotu bloku.

| **kódovaná hodnota** | **binární reprezentace** |
|---|---|
| -4 | 10000 |
| -3 | 10010 |
| -2 | 000 |
| -1 | 101 |
| 0 | 01 |
| 1 | 110 |
| 2 | 001 |
| 3 | 10011 |
| 4 | 10001 |
| ostatní | 111+8 bitů |

Tabulka 5: Kódování barevné hodnoty obrázků

Výsledek komprese je uložen do binárního souboru, kde je pro prvních 5 bajtů vyhrazen prostor na zaznamenání rozměrů a počtu obrázků. Kvůli přesně vymezenému prostoru pro tyto údaje je omezena velikost a počet obrázků. Prvních 14 bitů je vyhrazeno pro údaj o počtu obrázků z toho vyplývá, že maximální počet je 2 na 14 -1, což se rovná 16 383. Následujících 26 bajtů je vyhrazeno pro šířku a výšku. Maximální rozlišení tedy činí 8 191 x 8 191.

Následující prostor je už vyhrazen výhradně pro samotné kódování obrázků.

## 6.2 Dekodér

Na počátku dekomprese se nejprve načte informace o velikosti a počtu obrázků, které jsou důležité pro další průběh dekódování. Následně se již začne s dekomprimací a jako první se načte velikost prvního bloku. Ten se zapíše do bitmapy a následuje další blok, přičemž po každém uloženém bloku se kontroluje, jestli již nebyl zaplněn celý hlavní blok, aby se mohlo přejít k dalšímu obrázku.

Další důležitou úlohou dekodéru je, vždy umístit specifikovaný blok na správné místo v bitmapě. Tímto se docílí jednoznačně definovanému pořadí kódovaných bloků. Začíná se tradičně vlevo nahoře, následuje blok po pravé straně a nakonec spodní řada, která

se začíná ukládat také zleva. Takovéto uspořádání samozřejmě platí i v blocích o menší velikosti, kde přednost má vždy oblast o menší velikosti.

## 6.3 Dosažené výsledky

Při porovnání na základě stejné velikosti komprimované sady, dosahuje moje metoda podobných výsledků jako Dirac a Theora. Na obrázku č. 16 je vidět výřez jednoho z obrázků, který byl použit pro názornou ukázku kvality u jednotlivých kompresních metod. Sada která tento obrázek obsahuje má velikost 898 596 B a kompresní poměr 14,15. Na obrázku jsou patrné bloky různých velikostí jak byly popsány výše, ale původní stopa pro kterou byl testovací obrázek charakteristický je zachována.

Obrázek 16: Komprese s tolerancí rovné sedmi

Druhé testování bylo zaměřeno na porovnání velikosti při co možná nejvíce se přibližující identické kvalitě obrázků. V tomto testu se můj algoritmus s tolerancí rovné osmi, řadí na druhé místo spolu s VP8 za H.264. Velikost kolekce je v tomto nastavení 523 871 B a kompresní poměr je 24,28. Na přiloženém CD je v adresáři "Testování na základě stejné kvality"uložen jeden obrázek z této sady spolu s ostatními obrázky komprimovanými různými kompresními metodami.

Z hlediska časové náročnosti komprese a dekomprese se rychlost pohybuje od jedné až ke čtyřem minutám. Jelikož se jedná o návrh algoritmu, nejsou zde aplikovány optimalizace, jako například použití pointerů při práci s bitmapou. Při takové optimalizaci se předpokládá zlepšení o jeden řád. Další možné zlepšení by přineslo využití masivního paralelizmu na grafické kartě za použití technologií CUDA či OpenCL.

# 7 Závěr

První část práce byla věnována uvedení do problematiky kódování a seznámení s existujícími kompresními metodami. Důraz byl především kladen na ztrátové kompresní formáty videa, které jsou vhodné pro komprimaci velmi podobných obrázků.

Jedním z úkolů práce bylo najít vhodnou kompresní metodu pro dodanou sadu testovacích obrázků. Testování bylo provedeno dvěma způsoby. Prvním způsobem bylo porovnání na základě stejné velikosti komprimované sady a následné porovnání kvality. Druhá metoda spočívala v opačném pořadí. Prvně se stanovila přijatelná kvalita obrazu, do které se poté komprimovala sada testovacích obrázků. Porovnání bylo provedeno podle velikostí těchto sad obrázků. V obou testech nejvíce vynikal formát H.264 / MPEG-4 part 10, který ostatní testované metody předčil v kompresním poměru. Konkrétně tedy H.264 main profile, který měl o něco lepší výsledky než H.264 high profile.

Z hlediska rychlosti komprese dosahovali nejlepších výsledků formáty MPEG-2, MPEG-4 ASP, WMV1 a WMV2, u kterých se doba komprese pohybovala kolem 3,5 sekundy. Ovšem všechny tyto metody zaostávali ve výsledném kompresním poměru natolik, že nejsou vhodné pro kompresi takovéto sady podobných obrázků.

Dalším úkolem byla implementace vlastní varianty kompresního algoritmu, který byl naprogramován v programovací jazyku C# . Ačkoli tento algoritmus nepředčil standart H.264, dosáhl zajímavých výsledků a může být tak inspirací pro budoucí práce.

# 8 Reference

[1] MORKES, David. Komprimační a archivační programy. 1. vyd. Brno: Computer Press, 1998, 177 s. ISBN 80-722-6089-8.

[2] DAVID SALOMON, Giovanni Motta a With contributions by David BRYANT. Handbook of data compression. 5th ed. London: Springer, 2010. ISBN 978-184-8829-039.

[3] ČAPEK, Jan a Peter FABIÁN. Komprimace dat. Vyd. 1. Praha: Computer Press, 2000, viii, 173 s. Internet. ISBN 80-722-6231-9.

[4] SAYOOD, Khalid. Introduction to data compression. 3rd ed. San Francisco: Morgan Kaufmann Publishers, 2006. ISBN 01-262-0862-X.

[5] VEČERKA, Arnošt KOMPRESE DAT. Olomouc: KATEDRA INFORMATIKY 2008, PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITA PALACKÉHO

[6] HANKERSON, Darrel. Introduction to Information Theory and Data Compression. Boca Raton: Chapman, 2003, 366 s. ISBN 15-848-8313-8.

[7] PENNEBAKER, Khalid a Joan L MITCHELL. JPEG still image data compression standard. 3rd ed. New York: Van Nostrand Reinhold, 1992, xviii, 638 p. ISBN 04-420-1272-1.

[8] WATKINSON, John. The MPEG handbook: MPEG-1, MPEG-2, MPEG-4. Boston: Focal Press, 2001, ix, 395 p. ISBN 0 240 51656 7.

[9] MITCHELL, Joan L. MPEG video: compression standard. Chichester: Chapman, c1996, xxxv, 470 p. ISBN 04-120-8771-5.

[10] ARNOLD, J.F, M.R FRATER a J ZHANG. Error resilience in the MPEG-2 video coding standard for cell based networks – A review. Signal Processing: Image Communication. 1999, vol. 14, 6-8, s. 607-633. DOI: 10.1016/S0923-5965(98)00059-9.

[11] BHASKARAN, Vasudev a Konstantinos KONSTANTINIDES. Image and video compression standards. 2nd ed. Boston: Kluwer Academic Publishers, c1997, xvi, 454 s. Kluwer international series in engineering and computer science, SECS408. ISBN 07-923-9952-8.

[12] RICHARDSON, Iain E. The H.264 advanced video compression standards. Chichester: John Wiley, 2003, xxx, 316 s. ISBN 978-0-470-51692-8.

[13] PEREIRA, F a Touradj EBRAHIMI. The MPEG-4 book. 2nd ed. Upper Saddle River, NJ: Prentice Hall PTR, c2002, xxxvi, 849 p. ISBN 01-306-1621-4.

[14] Thomas Wiegand, Gary J. Sullivan, Gisle Bjontegaard and Ajay Luthra: Overview of the H264 / AVC Video Coding Standard, IEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, JULY 2003

[15] IEEE Transactions on Circuits and Systems for Video Technology. vol. 13, issue 7. ISSN 1051-8215. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1218195

[16] a Didier NICHOLSON. Video Compression Formats. Intelligent Video Surveillance Systems. Hoboken, NJ USA: John Wiley, 2012-12-17, s. 65. DOI: 10.1002/9781118577851.ch5. Dostupné z: http://doi.wiley.com/10.1002/9781118577851.ch5

[17] XIPH.ORG FOUNDATION. Theora Specification [online]. 16. březen 2011 [cit. 2014-03-15]. Dostupné z: http://www.theora.org/doc/Theora.pdf

[18] POYNTON, Charles A. Digital video and HD: algorithms and interfaces. 2nd ed. Waltham, MA: Morgan Kaufmann, c2012, xl, 707 p. ISBN 01-239-1926-6.

[19] BENNETT, J. a A. BOCK. In-depth Review of Advanced Coding Technologies for Low Bit Rate Broadcast Applications. SMPTE Motion Imaging Journal. 2004-12-01, vol. 113, issue 12. DOI: 10.5594/J16246. Dostupné z: http://journal.smpte.org/cgi/doi/10.5594/J16246

[20] (EDS)., Jianguo Zhang ... [et al.]). Intelligent video event analysis and understanding. 1. Ed. Berlin: Springer, 2011. ISBN 978-364-2175-534.

[21] WAGGONER, Ben a Ben WAGGONER. Compression for great video and audio: master tips and common sense. 2nd ed. Burlington, MA: Focal Press, c2010, xxxvi, 579 p., [16] p. of plates. ISBN 02-408-1213-1

[22] ARGUTEC, s.r.o. Argutec: Surface Scan System [online]. [cit. 2014-04-10]. Dostupné z: http://www.argutec.eu/surface-scan-system-cz

[23] BANKOSKI, J.; Wilkins, P.; Yaowu Xu; , "Technical overview of VP8, an open source video codec for the web,"Multimedia and Expo (ICME), 2011 IEEE International Conference on , vol., no., pp.1-6, 11-15 July 2011.

# A Obsah CD

Přiložené CD obsahuje tyto adresáře:

1. Zdrojový kód
2. Aplikace
3. Testovací sada obrázků
4. Testování na základě stejné kvality
5. Testování na základě stejné velikosti